September, 2017

Design Centric RPA Approach

The Jidoka® Example

Robotic Process Automation (RPA) is a revolutionary step forward in taking Enterprise business process optimization to the next level and in empowering 21st century working age with Digital assistants. As RPA market is going through its early growth phase, there are a variety of RPA product offerings that have emerged. From an implementation perspective, these can broadly be categorized into two distinct approaches which really determine the scope, development practices, skill requirements, roles and processes that shape up RPA implementations. This paper is focused on taking a deep dive into exploring one of these two approaches – The Design Centric Approach to RPA, and bringing out insights why it is important for complex strategic RPA implementations, what are the key aspects of this approach, and how to implement this approach by taking the example of Jidoka RPA product.

by **Deepak Sharma**

About the author

Deepak Sharma

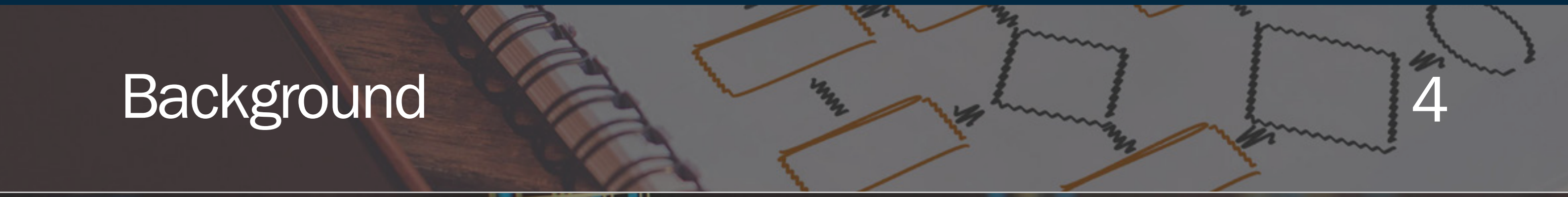
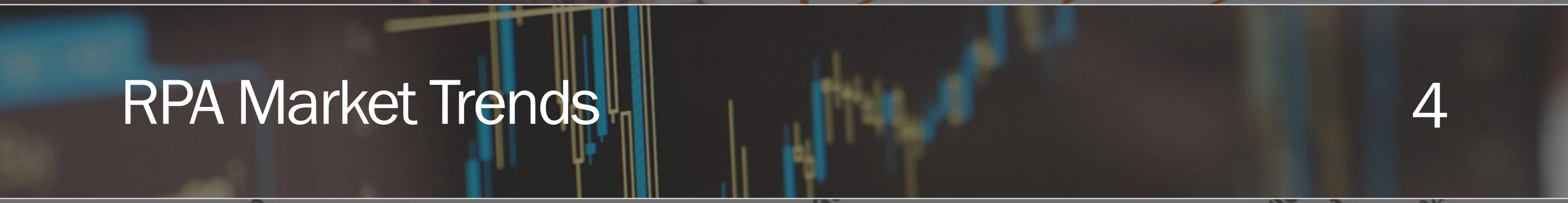
RPA Evangelist and Senior Automation Consultant

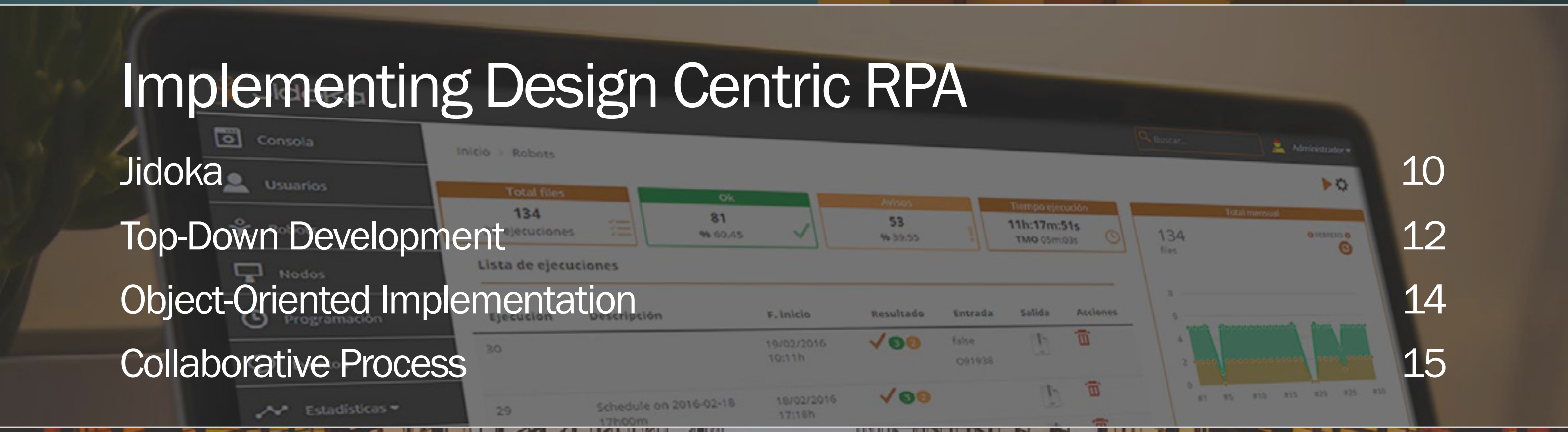
Deepak is a seasoned IT consultant with 15 years of experience in UI Automation for Test and RPA; focused on Enterprise Applications such as SAP, Siebel, PLM, Custom Wealth & Asset Management platforms, Healthcare Claim Processing systems, etc. He has worked with major global enterprises in the UK such as Barclays, IFDS, Department of Works and Pensions, RWE Npower, and in the US; such as Motorola & UnitedHealth Group. Deepak is passionate researcher, who writes about market trends, adoption challenges, and implementation approaches for the RPA Industry.

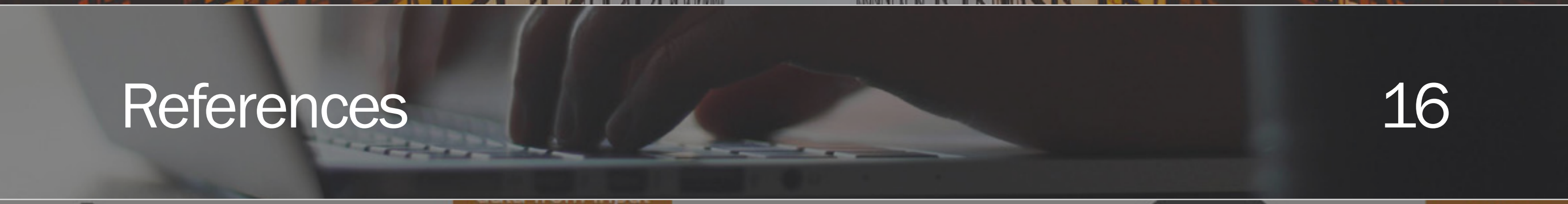
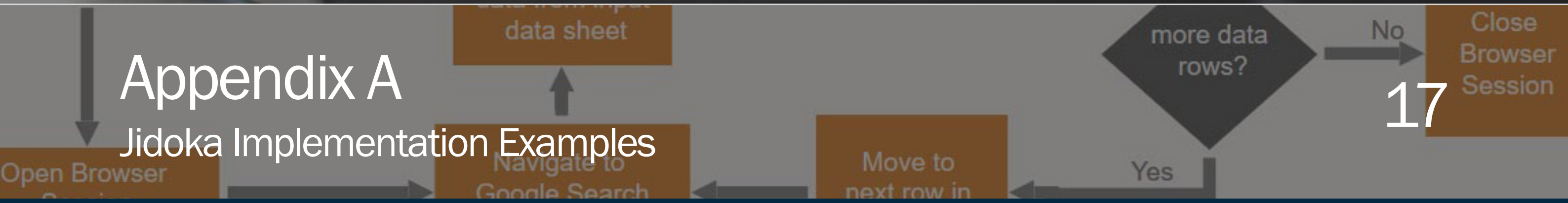
He is an evangelist of a Design Centric approach for RPA based on industry standard development practices and collaborative processes between operations and IT, which holds tremendous potential for enterprise customers to reach the next level in business process optimization and to act as a catalyst for digital transformation and business model innovation. Deepak holds a bachelor in Computer Science & Engineering degree and a Masters in Strategy & Innovation from the SAID Business School, University of Oxford where He secured a distinction in Innovation Strategy focused on market evolution and implementation of nascent technologies such as Robotics, 3D printing, AI and Big data.

About Jidoka®

Jidoka is a Robotic Process Automation (RPA) software that automates business processes replicating human behavior. Large companies from key economic sectors (banking, insurance, utilities, real estate, IT, BPO, among others) around the world utilize Jidoka's software robots to automate their front and back office processes, freeing people of tedious and repetitive tasks. Developed by **Novayre**, Jidoka holds a prominent position in the global RPA provider ecosystem delivering powerful software robots development tools combined with advanced enterprise level robot operations orchestration.



Background

I recently wrote an **article about the Record-Replay approach**[1] to RPA implementations and why it is not a preferred approach. I briefly touched upon the Design Centric Approach to RPA in that article. In this white paper, I would like to take a deep dive into exploring the Design centric approach to RPA, and bring out insights about some of the key issues faced during RPA implementations and why design centric approach is the right approach for strategic RPA implementations in complex enterprise environments. The latter part of this white paper demonstrates how the Design centric approach to RPA implementation can be applied in practical using the Jidoka RPA tool. And based on that, we would be reflecting on how the broad RPA implementation issues can be effectively resolved to develop and deploy highly efficient, high quality bots.

RPA Market Trends

Firstly, let's look at some of the broad trends and issues faced by the RPA market today as it is going through its **take-off phase**[2] and as many industry analysts are predicting- RPA market will continue to grow at a rapid pace for the next 5-7 years before it starts to level-off. As with the growth of any new technology market, there is a flood of different RPA product variants that have emerged with multitude of ideas about what the customers want. As more and more RPA products are getting pushed out into the market, it's increasingly the case that this does not create new products but instead has created new product categories such as pure-play RPA- attended or unattended, cognitive RPA, and AI powered RPA. Although, there are tall claims being made about RPA product offerings, to create a further **40-50% efficiency**[3] on the traditional BPO costs, the demand is still inchoate especially in the case of cognitive and AI powered where a lot of innovative activities is going on at the moment till the products mature.

Given all the hype, an ever-growing variety of products and categories on offer and an inchoate demand, the RPA market is going through its evolution with some products prospering and others falling apart. As more and more customers across industry sectors are trying out RPA products and conducting POCs, consumer needs are getting more articulated and specific. And as this continues to happen, the coming year or two will result in a sizeable shakeout to possibly reduce the number of providers and stabilize the RPA industry.

Why Design Centric Approach

Key Issues in RPA Implementations

There are three broad issues that we will focus on. **Firstly**, the biggest is the confusion over how to approach RPA implementations as there is a lot of misunderstanding in the market created by RPA products sales pitches. Should you go for a product that sound cheaper early days but offers quick, tactical POC outcomes automating simplistic processes using approaches that fall into the record-replay trap? Or should you go for a product that forces you to look at business process automation in a holistic way across a large operational area? This may sound higher investment and effort early days but so would be the resultant **RPA ROI**[4] and efficiency gains over time. It's a *tactical vs. strategic choice* for RPA implementation approach. As many of the implementation cycles are in early days and as more and more enterprise customers are trying to make sense of RPA benefits, many are struggling with the RPA product choice they've made and are finding it difficult to realize acceptable ROI from RPA implementation beyond the early POC stage.

Many consumers are struggling with the RPA product choice they've made and are finding it difficult to realize acceptable ROI.

Secondly, should you be developing bots that require very proprietary development & maintenance skills? Many RPA products are claiming to offer code free development of bots that the front-line operations users can carry out themselves using the pre-built capabilities offered in the tool. But seldom such RPA implementation tactics work and soon customers find themselves in a state where they have to bring in skilled RPA developers in the chosen tool from outside.

The situation is not helpful as many of the pureplay RPA tools don't necessarily provide or support standard software development kits (SDKs). As such, getting skilled software programmers, say in Microsoft or Java technologies for example, does not work. And finding resources locally that have been in operations and have become power user in proprietary features offered by an RPA tool is difficult. This problem is widespread for customers and one solution is to have their operations teams undergo training and acquire the necessary skills in the RPA tool they've chosen. But this doesn't provide implementation expertise to begin with, as needed for complex implementations, and is dependent on the availability of good quality training programs for the chosen tool. Instead, should you look to implement an RPA tool that applies industry standard development techniques for building software robots and does not create skills gap? It's a choice between *highly proprietary methodologies vs. industry standard development practices* for RPA implementations.

Many of the pureplay RPA tools don't provide or support standard Software Development Kits (SDKs).

Why Design Centric Approach - Key Issues on RPA Implementations

Thirdly, because of the implementation approach and development technique employed based the chosen RPA tool, should you push bot development fully into operations teams? Does RPA tools offer such silver bullet that RPA implementations be done without involvement from IT teams?

A direct consequence of this is the failure to scale RPA implementation efforts beyond the POC stage. Another, and on a more serious note, is to do with the level of technical quality of the solution in terms of robustness, modularity, security, scalability and maintainability. Or, should you look to adopt an RPA tool that inspires a collaborative effort between both the operations and the IT teams for building process bots? Thus, the resultant RPA implementation delivers not only the much-needed operational efficiency but is of high technical quality as well. In a technology world where collaborative approaches like Agile, DevOps are a new normal for Digital transformation with in businesses, it is not advisable to take a step backwards when it comes to implementing RPA and end up creating a silo between the operations and IT teams. It is a choice between a *collaborative vs. siloed approach* to RPA implementations.

The resultant RPA implementation delivers not only the needed operational efficiency but is of high technical quality as well.

The 3 broad issues facing RPA implementations

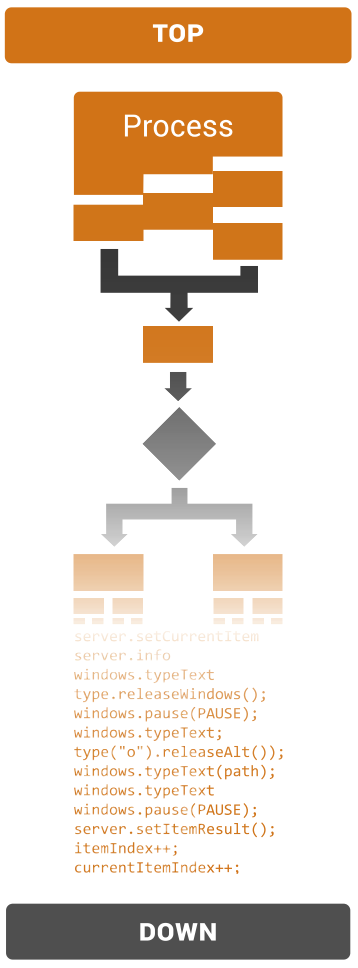
Problem	Root Cause	Solution
Low ROI Not taking a strategic approach to RPA implementation resulting in low ROI.	**Record Replay** **Bottom Up Tactical Approach**	**Design Centric** **Top Down Strategic Approach**
Skills Gap Not employing industry standard development techniques leading into skills gap.	**High Proprietary** **Bot Development Techniques**	**Standard Development** **Practises for Bot Development**
Failed Projects Not implementing RPA projects as a collaborative effort between operations and IT teams causing implementation failures with some industry analysts stating this to be as high as 60% in late 2016 - early 2017.	**Siloed Processes pushed torwards** **Operations away from IT**	**Collaborative process between Operations and IT**

Introduction to Design Centric RPA Approach

Let us now look at the Design Centric approach to RPA implementation and analyze how that helps to solve the broad problems facing the RPA industry that we've discussed above.

Top-Down Development | Object-Oriented Implementation | Collaborative Process

Design Centric approach to RPA implementations focuses on taking a **top-down approach**[5] for automating the business processes within an enterprise. It's about starting with designing *high-level workflow* for processes to be automated in a language that is understood by the domain experts, and not focusing straight into the technical implementation details by attempting to record an isolated bot for example in a bottom-up fashion. The processes to be automated are designed in their entirety by Operations SMEs. The high-level process flows are made deterministic to ensure all possible exceptions and recovery paths are covered. The whole idea follows the principle that Design is not coding and coding is not design.



```
server.setCurrentItem
server.info
windows.typeText
type.releaseWindows();
windows.pause(PAUSE);
windows.typeText;
type("o").releaseAlt());
windows.typeText(path);
windows.typeText
windows.pause(PAUSE);
server.setItemResult();
itemIndex++;
currentItemIndex++;
```

Design Patterns

The high-level process flows for automation are modeled using the standard design patterns for business process modelling. The process steps are considered as domain events and are labelled using a common language that describes the application domain. This lays the foundation for an object-oriented layered solution architecture.

Modularity

As business processes are modeled using a common domain language, the process steps that represent domain events once implemented in code provides a library of *automated reusable modules* or business objects that can be re-used across multiple automated process bots. This reduces the level of maintenance needed once automated process are released into production. Any change to a domain event can be made only in the corresponding automated module to reflect in all applicable bots.

Scalability

As processes within an operations area of a business are being automated and a library of domain specific automated process components starts to emerge, this provides for scalability in the RPA implementation to automate much greater number of processes within the enterprise with minimal additional effort.

Introduction to Design Centric RPA Approach

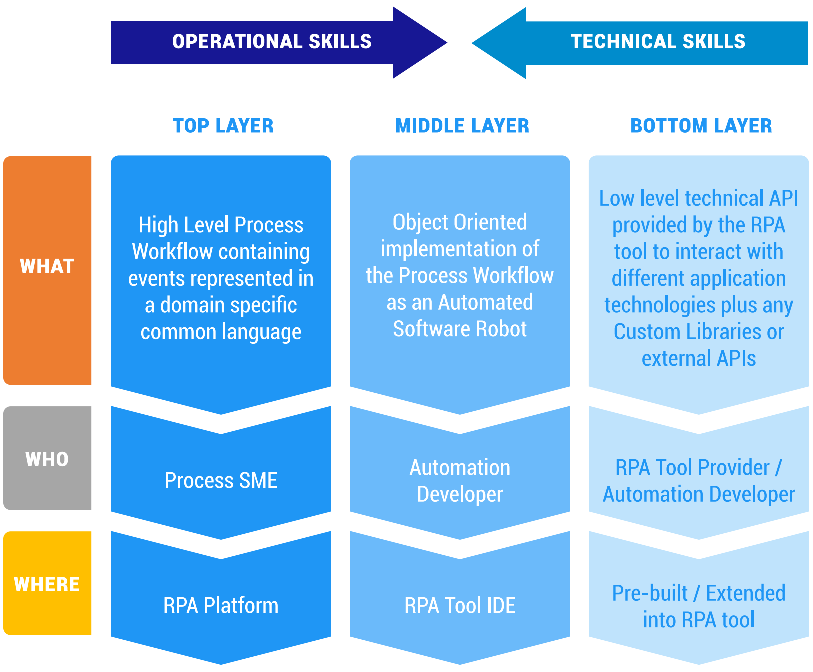
Let us now look at the Design Centric approach to RPA implementation and analyze how that helps to solve the broad problems facing the RPA industry that we've discussed above.

Top-Down Development | **Object-Oriented Implementation** | Collaborative Process

Implementation of automated process bots is done using industry standard *object-oriented programming* techniques. OOPS concepts are applied such as building objects to model application attributes and behavior as needed for RPA implementations. The technical implementation details are abstracted from the high-level process workflow. The technical details of object attributes (such as UI object locator attributes) are encapsulated from misuse and duplication. Inheritance is applied to implement more sophisticated custom actions extending the raw API calls provided by the RPA tool for complex implementation requirements.

Layered Architecture

The high-level process flow represents the top layer of the automated solution. The actual implementation of the process steps as a software robot is done in one or more bottom layers. Typically, the implementation of domain events as reusable process components is done in an application specific techno-functional middle layer that sits in between the business domain specific top layer and the technical base layer provided by the RPA tool. The base layer provides the technical API that the RPA tool uses to perform actions directly on the application/s under automation (for e.g. Web, Windows, Citrix, Excel, SAP, etc.). The diagram depicts the layered architecture of RPA solution.


OPERATIONAL SKILLS
TECHNICAL SKILLS
TOP LAYER
MIDDLE LAYER
BOTTOM LAYER
WHAT
High Level Process Workflow containing events represented in a domain specific common language
Object Oriented implementation of the Process Workflow as an Automated Software Robot
Low level technical API provided by the RPA tool to interact with different application technologies plus any Custom Libraries or external APIs
WHO
Process SME
Automation Developer
RPA Tool Provider / Automation Developer
WHERE
RPA Platform
RPA Tool IDE
Pre-built / Extended into RPA tool

Extensibility

The use of standard development technologies allows for extensibility to utilize proven external APIs or libraries, or by building custom libraries to enhance the existing solution and fulfil specific technical implementation requirements.

Introduction to Design Centric RPA Approach

Let us now look at the Design Centric approach to RPA implementation and analyze how that helps to solve the broad problems facing the RPA industry that we've discussed above.

Top-Down Development | Object-Oriented Implementation | **Collaborative Process**

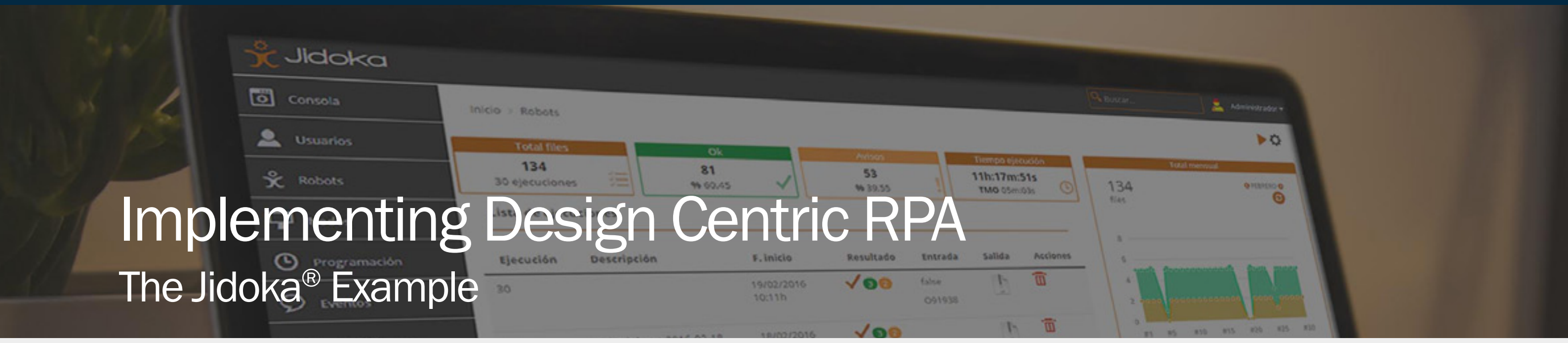
The business process to be automated is first designed as a high-level process workflow by one or more process SMEs, who are experts in the given operational area within the enterprise. The process workflow is created in the workflow designer provided by the RPA platform. An automated bot is then created by implementing the domain events in the process workflow as object-oriented code. This is done by one or more automation developers in the IDE provided by the RPA tool, decoupled from the high-level process flow.

The automated implementation in the form of object libraries and methods are then published to the RPA platform. Process SMEs are then able to map the domain events in the process flow to the automated components, test the automated process and release it into production. Both, process SMEs and automation developers, work in collaboration to deliver the automated process bots. The automation developer ensures the technical quality of the solution whilst the process SME ensures that the automated process bots are accurately designed to deliver true operational efficiency to the business.

Automation Developers

+

Process SMEs



Implementing Design Centric RPA

The Jidoka® Example

Let's now shift our attention to show how the design centric approach we've described above can be put into practice by taking the example of Jidoka RPA tool. And towards the end, we will assess how this approach allows to effectively resolve the broad RPA implementation issues that we discussed earlier.

But firstly, let me provide a brief about the Jidoka RPA tool and explain why this tool provides a great fit to showcase the Design Centric implementation approach for RPA.

Jidoka is a leading RPA solution in the RPA Spanish-speaking market, developed by **Novayre** a technology-based company that specializes in software development, integration and automation. The vendor talks about many key technological differentiators in their product catalogues but, having tried the tool hand on, the ones that stood out for me are:

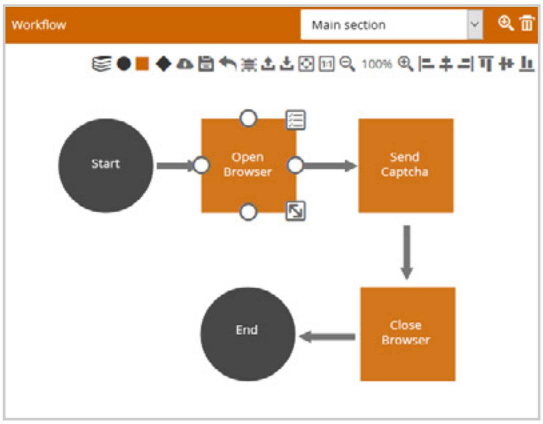
Development Technology

It's *Java*! Unlike many other Microsoft based Core RPA tools, Jidoka is a 100% Java platform providing a more open, universal character and an easier adaptability to the most complex business processes and environments. Jidoka is like the Selenium of RPA World!

Jidoka is a 100% Java platform providing an easier adaptability to the most complex process.

Workflow Designer

For domain SMEs, Jidoka platform provides a visually attractive and easy to use *console* for modelling complex business processes to begin the automation process separate from the SDK.



The Jidoka platform provides a visually attractive console for modelling business process.

REST API

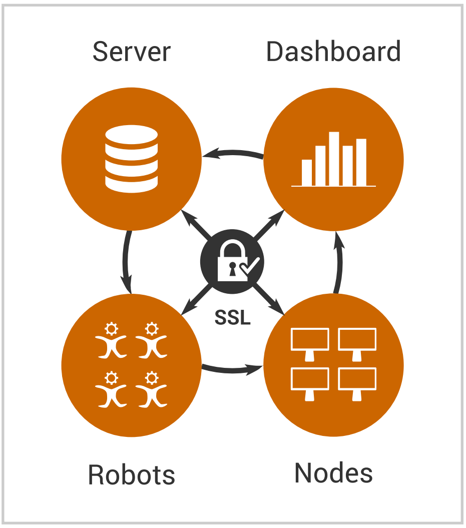
Jidoka Server provides a *REST* (Web Services) API that allows Jidoka to integrate with other Intelligent automation tooling and to interconnect with other systems in highly complex enterprise operating environments.

Implementing Design Centric RPA - The Jidoka® Example

Solution Architecture

The only pureplay RPA tool that I found so far to have ticked all the components that I mentioned in **my generic RPA solution architecture**[6].

Server Platform	Provides standard features such as central storage of robots and library components, orchestration of robots on client nodes, control panel that gives access to robots and nodes configuration, dashboard that provides analytics information, central store of execution logs.
SDK Tool	If you are one among the **9 million Java developers**[7], you can implement software robots using Jidoka in a software development kit (SDK) of choice, for instance Eclipse, and can enhance your robots with any proven Java libraries available from popular internet sources.
Client Nodes	Jidoka nodes can be installed on machines with any operating system that supports Java Virtual Machine (JVM), even Linux.
Configuration Management	And yes! You can also *GIT* your robot code. Jidoka does not restrict you which SDK and what source code tool you want to use, distributed or centralized, within your team.
Supported Application Technologies	Core Jidoka API provides modules to automate standard technologies such as Web, Windows, MS Office, databases, file sources, etc. Additionally, Jidoka has successfully demonstrated in a number of high profile enterprise projects over the years the ability to develop your own custom libraries for various ERP and CRM technologies such as SAP, Oracle, terminal emulators, etc.



The platform provides a central storage of robots and library components and a dashboard that provides analytics information.

Implementing Design Centric RPA

Let us now look at how the key aspects of a Design centric approach to RPA implementation can be applied using the Jidoka® tool by taking a simplistic example of a software robot.

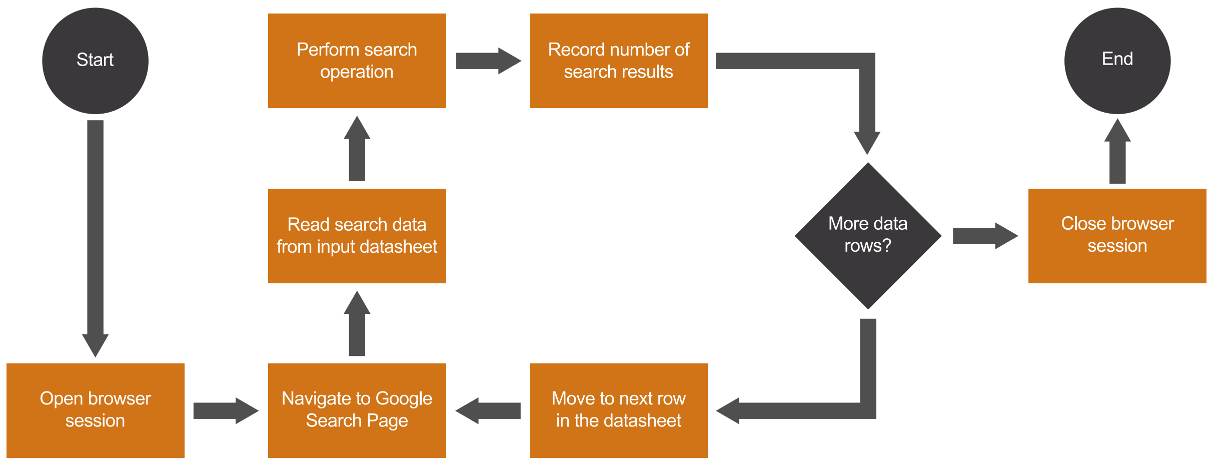
Complex real life Jidoka implementation examples in **Appendix A.**

Top-Down Development | Object-Oriented Implementation | Collaborative Process

Development of a RPA bot is guided by a business process workflow that is created in the Jidoka platform console as the first thing. This is done by modelling the set of actions and transitions that apply within a process as can be seen in the example as below.

The steps in the process workflow are kept as high level process events only and are labelled using a common language that the operations SMEs can easily define and maintain for the application domain in context. The example here shows a robot that is designed to open a browser session, navigate to a web application, read data from an excel spreadsheet, perform an operation, capture the output of the operation, and repeat these steps till all the data iterations are complete.

The high-level domain events in the process workflow once implemented in code provides for a library of automated process components that can be reused across multiple process bots reducing the maintenance effort across the RPA implementation. As the library of domain specific automated components gets established, it provides for scalability to implement further bots within that operations area of the business.


Start
Open browser session
Navigate to Google Search Page
Read search data from input datasheet
Perform search operation
Record number of search results
More data rows?
Move to next row in the datasheet
Close browser session
End

Implementing Design Centric RPA

Let us now look at how the key aspects of a Design centric approach to RPA implementation can be applied using the Jidoka® tool by taking a simplistic example of a software robot.

Complex real life Jidoka implementation examples in **Appendix A.**

Top-Down Development

Object-Oriented Implementation

Collaborative Process

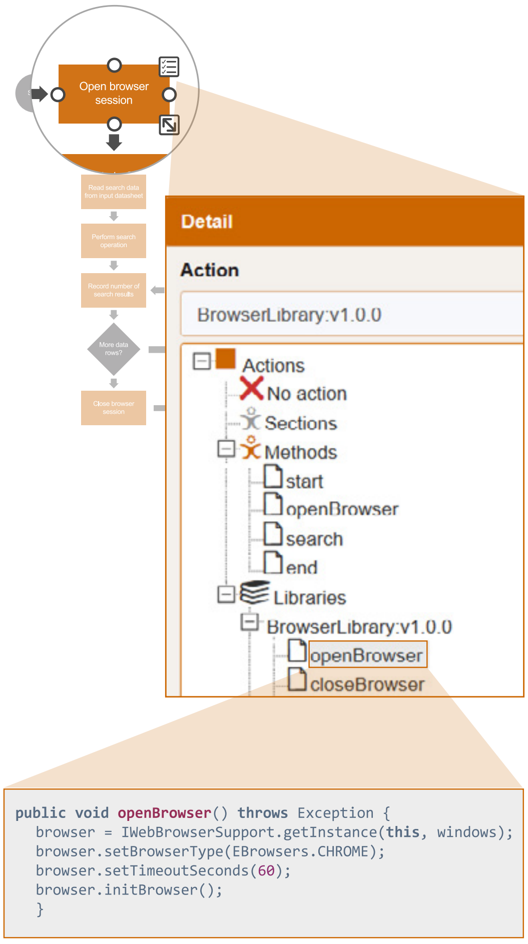
As Jidoka uses Java, the implementation of the high-level process workflows as automated bots is done by applying object oriented programming concepts. Classes & objects are developed to capture the exact details of the application UI components as attributes and the implementation behaviour of the process events as methods. The technical implementation details as such are abstracted from the high-level process workflow visible in the console. Given Jidoka uses Java programming language, OOP concepts such as inheritance are applied extensively to develop more sophisticated custom actions by extending the core API that the tool provides.

Once the implementation code has been developed in a Java SDK of choice, it is published on to the Jidoka server using Maven to become available in the platform console as methods and libraries. These application specific modules available as Java methods can then be mapped to the high-level process steps in the workflow as shown in the example below.

As you can see by expanding the Open Browser Session process node from the workflow, all the published Java methods and libraries are visible to choose from and the "openBrowser" method is mapped to that process node.

The automated bots are implemented using a layered solution architecture. The high-level process flow represents the top layer of the automated solution. The step by step implementation code as Java methods & libraries represents the application specific middle layer of the automated solution. The core technical API layer of the Jidoka tool acts as the base layer providing different modules to automate against various types of standard UI interfaces such as Web, Windows, Excel, etc.

The middle layer methods are implemented by making the base layer API calls as shown in the example. Here the browser object is created by calling the "IWebBrowserSupport" interface which is part of the core browser module provided by the Jidoka tool.



```
public void openBrowser() throws Exception {
   browser = IWebBrowserSupport.getInstance(this, windows);
   browser.setBrowserType(EBrowsers.CHROME);
   browser.setTimeoutSeconds(60);
   browser.initBrowser();
   }
```

Implementing Design Centric RPA

Let us now look at how the key aspects of a Design centric approach to RPA implementation can be applied using the Jidoka® tool by taking a simplistic example of a software robot.

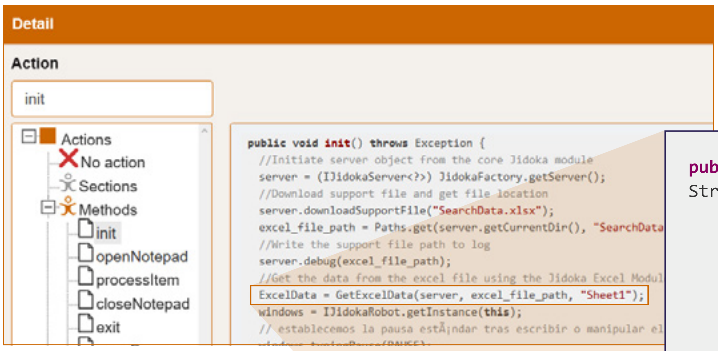
Complex real life Jidoka implementation examples in **Appendix A.**

Top-Down Development | **Object-Oriented Implementation** | Collaborative Process

Given Jidoka uses Java to develop bots, the automated solution is highly extensible in complex situations to allow use of proven external Java APIs into the automation projects and for building custom methods and libraries. The below example shows a custom library function that makes use of Apache POI, a standard Java API for working with Microsoft Documents, to read data from an excel file and provide it to the robot in the form of a two-dimensional array.

The custom library function “GetExcelData” in this example robot is being called inside the “init” method, as shown below, which is mapped to the “Start” node of the process workflow. This gives the bot the ability to read the execution data at the start step in this example and then use it through the bot execution.

Please note that Jidoka tool does provide a core API module to work with Excel files and this example to use Apache POI as an external API is for illustrative purpose only.



```
public void init() throws Exception {
  //Initiate server object from the core Jidoka module
  server = (IJidokaServer<?>) JidokaFactory.getServer();
  //Download support file and get file location
  server.downloadSupportFile("SearchData.xlsx");
  excel_file_path = Paths.get(server.getCurrentDir(), "SearchData
  //Write the support file path to log
  server.debug(excel_file_path);
  //Get the data from the excel file using the Jidoka Excel Modul
  ExcelData = GetExcelData(server, excel_file_path, "Sheet1");
  windows = IJidokaRobot.getInstance(this);
  // establecemos la pausa estÃ¡ndar tras escribir o manipular el
```

CONSOLE

ECLIPSE

```
public static String[][] GetExcelData(IJidokaServer<?> server,
String ExcelFileLoc, String SheetName) throws IOException

    {
    FileInputStream fis = new FileInputStream(ExcelFileLoc);
    XSSFWorkBook wb = new XSSWorkbook(fis);
    XSSFSheet ws = wb.getSheet(SheetName);
    int rowcnt = ws.getLastRowNum()+1;
    rowcount = rowcnt;
    int colcnt = ws.getRow(0).getLastRowNum()+1;
    server.debug(rowcnt+", "+colcnt);
    String[][] ExcelData = new String[rowcnt][colcnt];
    String value;
    for (int i=0; i<rowcnt; i++) {
        XSSFRow row = ws.getRow(i);
        for (int j=0; j<colcnt; j++) {
            XSSFCell cell = row.getCell(j);
            value = celltoString(cell);
            server.debug(value);
            ExcelData[i][j] = value;
        }
    }
    return ExcelData;
    }
```

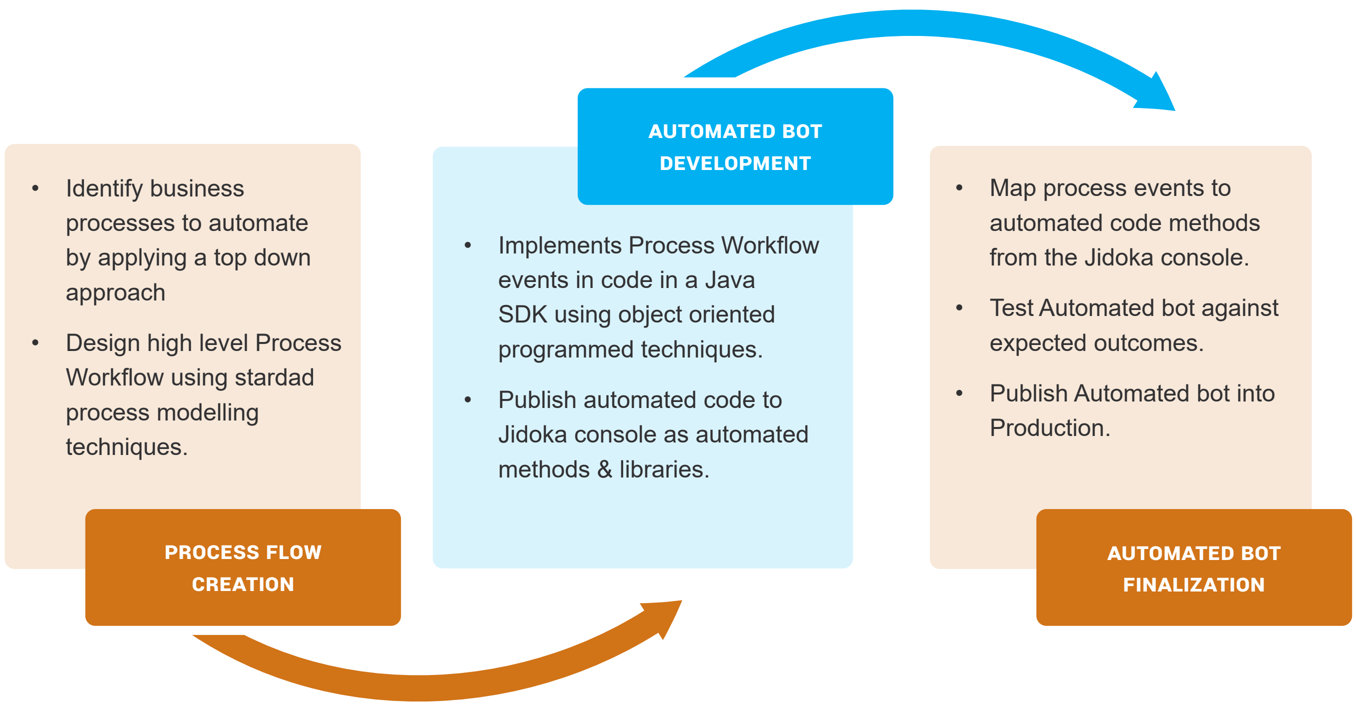
Implementing Design Centric RPA

Let us now look at how the key aspects of a Design centric approach to RPA implementation can be applied using the Jidoka® tool by taking a simplistic example of a software robot.

Complex real life Jidoka implementation examples in **Appendix A.**

Top-Down Development | Object-Oriented Implementation | **Collaborative Process**

Jidoka tool enables a collaborative development effort between the Process SMEs and the Automation Developers as depicted in the below process flow. As you can see, Jidoka tool provides a truly collaborative role-based automation implementation process, one that leads to high operational efficiency and high technical quality of the software bots produced.



PROCESS FLOW CREATION

- Identify business processes to automate by applying a top down approach
- Design high level Process Workflow using stardad process modelling techniques.

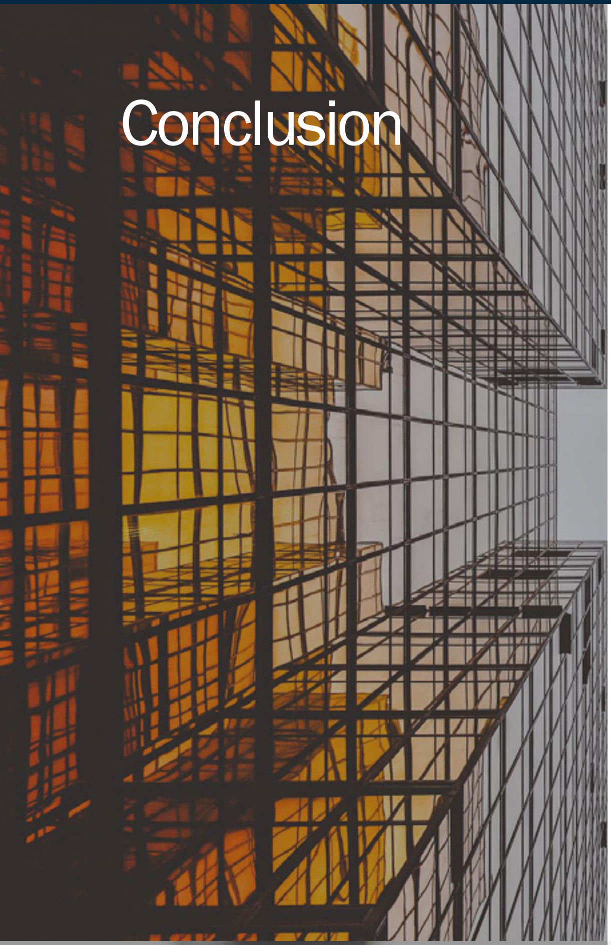
AUTOMATED BOT DEVELOPMENT

- Implements Process Workflow events in code in a Java SDK using object oriented programmed techniques.
- Publish automated code to Jidoka console as automated methods & libraries.

AUTOMATED BOT FINALIZATION

- Map process events to automated code methods from the Jidoka console.
- Test Automated bot against expected outcomes.
- Publish Automated bot into Production.

Process SME | Automation Developer



Conclusion

To conclude, let us look back at the broad issues we discussed in the first part of this article and assess how a design centric RPA implementation approach helps solve those issues. Firstly, taking a top-down strategic approach to RPA implementation to begin with designing the process workflows first, as in case of Jidoka using the platform console, provides a highly efficient and modular approach for automating complex business processes giving scalability to RPA implementations resulting in high ROI. Secondly, by employing industry standard object oriented implementation techniques offers a solid technical foundation, layered architecture and extensibility to the RPA solution for developing high quality bots. This offers a pragmatic solution to the resourcing issues in the RPA industry to tap into the existing extensive development community, such as for Java technology in the case of Jidoka. And thirdly, by implementing RPA projects as a collaborative effort between the operations and the IT teams with a clear role-based development process, as depicted in case of Jidoka, significantly reduces the risk of failure in RPA implementation projects caused either due to technical or operational holes in the bots.

References

↑ [1] **Sharma, D. (2017, March 23).**
Record Replay RPA - A boon or a bane?

↑ [2] **Press, G. (2017, January 23).**
Top 10 Hot Artificial Intelligence (AI) Technologies.

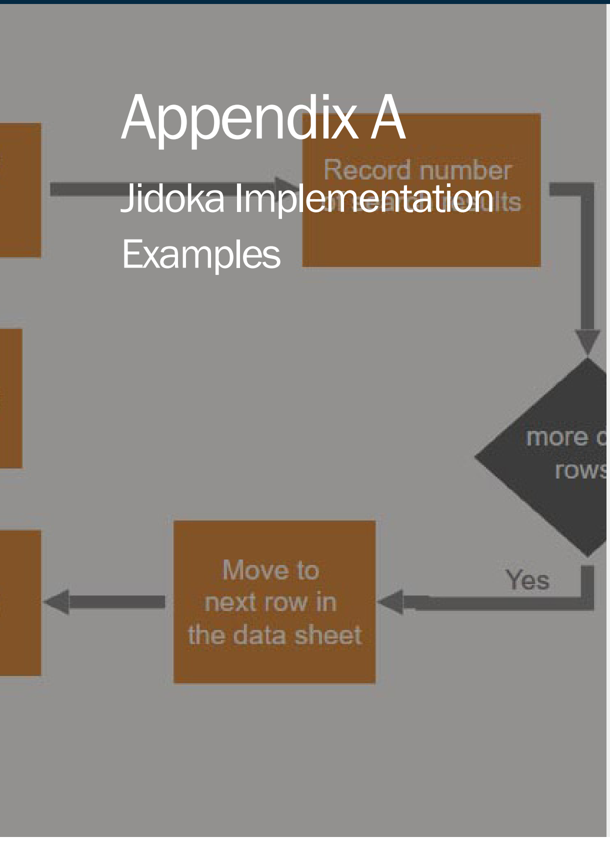
↑ [3] **Casale, F. (2015).**
Introduction to Robotic Process Automation A Primer.

↑ [4] **Sharma, D. (2017, March 2).**
Analysing impact on ROI from RPA AI implementations using System Dynamics Approach.

↑ [5] **Ayllón, V. (2016).**
Record or develop software robots? That's the issue in RPA.

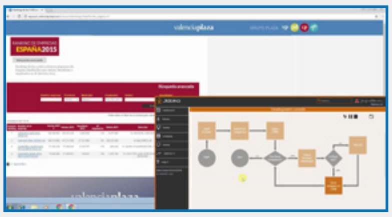
↑ [6] **Sharma, D. (2017, January 12).**
RPA Solution Architecture.

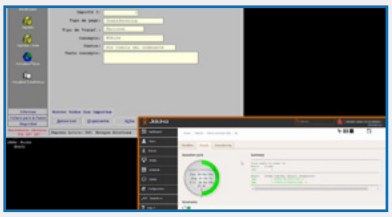
↑ [7] **Beneke, T., & Wieldt, T. (2013).**
JavaOne 2013 Review: Java Takes on the Internet of Things.



Appendix A

Jidoka Implementation Examples


Searching for potential clients in CRM Application


Payroll Automation